Provable Security for Outsourcing Database Operations

Sergei Evdokimov, Matthias Fischmann, Oliver Günther
Humboldt-Universität zu Berlin
{evdokim,fis,guenther}@wiwi.hu-berlin.de

Database outsourcing, whilst becoming more popular in recent years, is creating substantial security and privacy risks. In this paper, we assess cryptographic solutions to the problem that some client party (Alex) wants to outsource database operations on sensitive data sets to a service provider (Eve) without having to trust her. Contracts are an option, but for various reasons their effectiveness is limited [2]. Alex would rather like to use *privacy homomorphisms* [6], i.e., encryption schemes that transform relational data sets and queries into ciphertext such that (*i*) the data is *securely hidden* from Eve; and (*ii*) Eve computes hidden results from hidden queries that Alex can *efficiently* decrypt. Unfortunately, all privacy homomorphisms we know of lack a rigorous security analysis. Before they can be used in practice, we need formal definitions that are both sound and practical to assess their effectiveness.

The strongest notion of confidentiality requires every single bit of information on queries as well as data itself to be kept secret from Eve, no matter how many queries she can observe, or how rudely she breaks the contract. This notion would allow for doing business with arbitrarily malicious service providers – but is impossible to achieve. We present and analyse a solution for the case that (*i*) Alex trusts Eve to behave according to protocol and not secretly analyse the data she can gather, but (*ii*) is worried about Eve's data being sold to an adversary of Alex.

Related Work: In 2002, Hacıgümüş et al. propose a database encryption scheme for full SQL [4]. The idea is that every tuple is encrypted with a secure cipher first, then weakly encrypted attributes are attached to the ciphertext. These weak encryptions are obtained by taking a plaintext attribute value, mapping it to a containing interval, and encrypting that interval using a secret permutation. Two plaintexts may therefore map to the same ciphertext, even if they are not equal. Some of the information contained in the plaintext is destroyed but not as much as in an ordinary encryption scheme. While the remaining information (like the number of tuples of the table, or which tuples have similar values in which secret attributes) is enough to query the encrypted database with little post-processing, one hopes that Eve still does not obtain *too much* information on the data if the partitionings into intervals and the other security parameters have been chosen properly. However, the security is based on intuition alone; a rigorous analysis is missing.

1 Relevant Notions of Security

An encryption scheme is a tuple $(\mathcal{K}, E, D)$, where $E : \mathcal{K} \times X \mapsto Y, D : \mathcal{K} \times Y \mapsto X$ are encryption and decryption functions that convert a key and plaintext $x \in X$ into the corresponding ciphertext $y \in Y$. It must hold that $D_k(E_k(x)) = x$, or for short: $D(E(x)) = x$. Keys are chosen uniformly at random from the key space $\mathcal{K}$. The bit length $n = \log(|\mathcal{K}|)$ of all keys is called the *security parameter* of the scheme. We now introduce a class of encryption schemes called database privacy homomorphisms that has additional properties:

Definition 1.1 (Database PH) *Let* $(\mathcal{K}, E, D)$, $(\mathcal{K}, E^q, D^q)$ *be encryption schemes,* $\mathbb{R}$ *be a set of relations,* $\mathbb{C}$ *be a set of ciphertexts. A* database privacy homomorphism *(or* database PH*, for short) is a tuple* $(\mathcal{K}, E, E^q, D)$ *such that*

1. $E : K \times \mathbb{R} \to \mathbb{C}$ *encrypts tables,*
   $D : K \times \mathbb{C} \to \mathbb{R}$ *decrypts tables, and*
   $E^q : K \times \{\sigma_i\} \to \{\psi_i\}$ *encrypts queries.*
2. *For any relation* R *and any relational operation* σ_i, $E_k(\sigma_i(R)) = \psi_i(E_k(R))$.

Intuitively, that means that the plaintext operations can be carried out on the ciphertext, producing ciphertexts containing the result. Note that our definition contains an extra query encryption function E^q. This turns out to be useful for reasoning about data confidentiality: In some attacks, if Eve knows a few plain-

Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)
8-7695-2570-9/06 $20.00 © 2006 IEEE

text queries and some matching encrypted data, she can infer significant information on the plaintext data.

Also, we only consider schemes that perform tuple-by-tuple encryption, i.e., for table $R = \{v_1, \ldots, v_n\}$, the ciphertext is the set $C = \{c_1, \ldots, c_n\}$, where c_i is the encryption of tuple v_i.

The most common notions of security are based on games between an honest user (such as Alex) and an adversary (such as Eve) modeled by a probabilistic polynomial-time algorithm. A scheme is secure if Eve cannot win the game with probability $\frac{1}{2} + 1/p(n)$ for *every* polynomial p in polynomial time for sufficiently large n. If this is true, the winning probability is called *negligible*. The following (or similar) definition can be found in any textbook on cryptography.:

Definition 1.2 *An encryption scheme* $(\mathcal{K}, E, D)$ *is* indistinguishably secure *if Eve cannot win the following game:*

1. *Eve chooses two plaintexts* m_1, m_2 *of the same length and presents them to Alex.*
2. *Alex chooses* $i \in \{1, 2\}$ *uniformly at random and presents* $E_k(m_i)$ *to Eve.*
3. *Eve must guess* i*.*

E.g., Eve can win this game if by having plaintext encrypted she can learn enough about it to decrypt one bit of a chosen ciphertext. She merely needs to choose the plaintexts such that this bit suffices to distinguish the two.

This definition considers a passive adversary that only intercepts and processes the ciphertexts. There are exist more powerful adversary models capable of producing plaintexts and ciphertexts using encryption and decryption oracles.[1] But it is already easy to find adversaries that win the game in this weak sense for most schemes. Consider the scheme proposed by Hacıgümüş et al. in [4]. Let Eve produce two tables:

table 1:

ID	salary
171	4900
481	1200

table 2:

ID	salary
171	4900
481	4900

Next, Eve obtains a ciphertext, which is the encryption of one of the tables, from Alex. According to the scheme, the salaries in the first table will be mapped to different intervals with high probability. The salaries in the second table will be mapped to the same interval. Since the intervals are encrypted deterministically, the weak encryptions of the "salary" attribute of the first table will different, and the analogous weak encryption for the second table will be identical. Hence, Eve can determine with high probability to which table corresponds the received ciphertext: If there are two different weak encryptions of the "salary" attribute, Eve outputs 1; otherwise, she outputs 2. Similar attacks work on the scheme of Damiani et al. [3].

[1] Eve is said to be an active adversary if she can produce samples of her choice. This is modelled with a fictitious *oracle*. In practice, Eve usually obtains her answers from Alex by sending him confusing messages. If Alex is not a human but a machine running a complex application, this is a very real risk.

2 Database Privacy Homomorphisms and Security Limitations

Although privacy homomorphisms are just a subset of the set of all encryption schemes, the ability to transform plaintext operations into corresponding ciphertext operations yields new possibilities for an adversary to get insights into encrypted data. Consider an indistinguishably secure database PH. If an adversary runs the query $\sigma_{a_i=d}$ on the encrypted table, she receives a set of encrypted tuples. Though she cannot decrypt them, she knows that the value of the attribute a_i of these tuples is d.

Why does a scheme that is secure against the strongest adversary still leak so much information? The problem is that the classical model considers only parties operating with plain- and ciphertexts. But in this scenario parties also exchange queries and the results of these queries which can reveal information about plaintext. Kantarcıoğlu and Clifton [5] propose a definition that addresses this problem in which they state that a database PH is secure if any two tables with the same number of tuples as well as any two queries returning the same number of tuples are indistinguishable. They suspect that it may already be too strong to have a solution. Although we show that this is not true, there is another problem: A scheme secure in this sense does allow the adversary to get information about the plaintext with high probability.

Consider an example where Alex owns a database with statistics for three competing hospitals, keeping track of the state in which patients are leaving each hospital. Each patient is described by the attributes *id*, *name*, *hospital*, and *outcome* (*outcome* is a binary attribute either set to 'fatal' or 'healthy'). Now suppose that Eve knows the database schema, the number of hospitals, and has good estimates of the distribution of patient flows $(0.2, 0.3, 0.5$ resp.) and the ratio of fatal vs. successful outcomes $(0.08, 0.92)$.

Alex issues the following queries:

```
SELECT * FROM table WHERE hospital = 1;
SELECT * FROM table WHERE hospital = 2;
SELECT * FROM table WHERE hospital = 3;
```


Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)
8-7695-2570-9/06 $20.00 © 2006 IEEE

```
SELECT * FROM table WHERE outcome = 'fatal';
```

From the size of the results and the fact that we only have exact selects, Eve can guess the exact queries with high confidence (conditions on *id* and *name* yield far fewer hits). Then, by intersecting the answers to the first and the fourth query, Eve can infer the ratio of lethal to successful outcomes in hospital 1!

In another example we consider an active adversary with the ability to get encrypted queries of her choice and show that no matter how secure the table is encrypted, such an adversary is able to deduce a significant amount of information about the encrypted data. Suppose there was a patient "John" and Eve wants to find out in which hospital he was treated and what happened to him. She issues the encryption of query $\sigma_{name:John}$ using the query encryption oracle. Then Eve issues encryptions of queries $\sigma_{hospital:X}$, $X \in \{1,2,3\}$. By intersecting the results of the four queries issued, Eve can determine the hospital where John was treated. Analogously, she can find his status.

The definition of a secure database PH addressing these issues can be formulated as follows:

Definition 2.1 *A database PH provides indistinguishability if Eve cannot win the following game:*

1. *Eve chooses two tables $T_1(R), T_2(R)$ containing the same numbers of tuples and presents them to Alex.*
2. *Alex chooses $i \in \{1,2\}$ uniformly at random and presents $E_k(T_i(R))$ to Eve.*
3. *Eve receives at most q encrypted queries issued to $E_k(T_i(R))$ and computes the results (in case of active adversary Eve has access to the queries encryption oracle and issues q encryptions of plaintext queries of her choice).*
4. *Eve must guess i.*

The above examples demonstrate that this level of security cannot be achieved by *any* database PH, no matter how advanced. This can be established formally:

Theorem 2.1 *Any database PH $(\mathcal{K}, E, E^q, D)$ is insecure in the sense of Definition 2.1 if $q > 0$.*

Although if the adversary is passive, the case is less obvious, in both cases the security of the encrypted data cannot be guaranteed.

Intuition indicates that Definition 2.1 is too strong. Several relaxations can be considered: (*i*) Allow passive adversaries only, (*ii*) allow for some limited information leakage, or (*iii*) grant less plaintext knowledge to Eve. Unfortunately, none of these ideas is very promising.

(*i*) Passive adversaries are indeed an option in general. However in our setting, Eve has unlimited write access to all ciphertext and can produce any message sequence necessary for her attack. To demonstrate the plausibility of active adversaries that have considerably fewer options than Eve, Bleichenbacher has implemented a practical attack against SSL in which the role of the oracle is played by a confused web server [1]. (*ii*) Often only one bit of information is of interest to Eve (e.g., in the acknowledgement resp. cancelling of a stock order sent to a broker). Even if this is not the case, it is usually infeasible for the application designer to specify which bits are confidential and which are not. Hence, a secure scheme must not leak a single bit. (*iii*) Changing the amount of Eve's plaintext knowledge is the most problematic approach. If an encryption scheme is used as a building block for building complex applications, Eve must be assumed to have full access to the source code. In combination with timing attacks and other traffic analysis techniques, this gives her extensive knowledge on the structure and purpose of the encrypted data. Assuming a less well-informed Eve is a textbook case of *security by obscurity* and usually leads to disaster. The Risk Digest provides a rich collection of anecdotical evidence: `http://catless.ncl.ac.uk/Risks`. For a more detailed discussion, see the full version of this paper.

So what can be done? Assume that Alex trusts Eve not to attack him directly but still worries about her becoming adversarial in the future (e.g., by a change of company ownership). If Alex's trust in Eve deteriorates, he can cancel the contract in time and stop sending queries. Consequently, $q = 0$ and Theorem 2.1 does not apply. In the next section we present a solution for this case.

3 A Privacy Homomorphism Preserving Exact Selects

To conclude this poster, we give a general construction of a database PH based on searchable encryption schemes. One such scheme has been proposed by Song et al. [7] and includes a proof of security, but others can be used instead. In the full version of this paper, we describe a few straight-forward optimizations such as attributes of variable length, and present a formal security proof of our construction under the assumption that the underlying searchable encryption scheme is secure.

We write $a|b$ for concatenation of strings a and b. `'#'` is the padding symbol. Consider

Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)
8-7695-2570-9/06 $20.00 © 2006 IEEE

the relation `Emp(name:string[9], dept:string[5], salary:int)`. We can construct a database privacy homomorphism for the schema as follows.

First, create *words* that are strings of the same length (the shorter the better) and that identify the attributes of the relation. Then bijectively map the tuples of the given relation to the *documents*, or sets of words.[2] The number of words in each document is equal to the number of the attributes in the relation. The globally fixed word length is the length of the longest attribute value plus the length of an attribute identifier (required for decryption, see below). For example, the relation `Emp` will be mapped to the following set of documents: {`string[9]`|"N", `string[9]`|"D", `string[9]`|"S"}.

Now, map all tuples of the original relation `Emp` to documents. For example:

$$\langle \texttt{name:"Montgomery"}, \texttt{dept:"HR"}, \texttt{sal:7500} \rangle \mapsto$$
$$\{\texttt{"MontgomeryN"}, \texttt{"HR########D"}, \texttt{"7500######S"}\}$$

These documents are encrypted using a searchable encryption scheme and stored on a remote server.

In order to perform exact selects on the encrypted relation, queries of

$$\sigma_{attribute\_name:value}$$

will be mapped to the search operation

$$\varphi_{\texttt{toString}(attribute\_value)|"attribute\_id"}$$

and processed as a search operation, returning a set of encrypted strings. The strings then are decrypted and mapped to the corresponding tuples, producing the result of the issued query. E.g.:

$$\sigma_{name:"\text{Montgomery}"} \mapsto \varphi_{"\text{MontgomeryN}"}$$

Note that some searchable encryption schemes, and in particular the scheme presented in [7], sometimes return false positives. Alex needs to run a filter on the output. As the error rate is relatively small for all practical purposes, this does not affect the efficiency of our construction.

4 Conclusions and Future Work

If a database production system is deployed, nobody questions the virtue of a sound theory of databases that this system is based on. If an insecure network connection is used between client and server for sensitive applications, reliable encryption and authentication mechanisms are used to protect the user against attacks. However, for the case that the database server itself goes adversarial, recent work has been based on much lower standards: Rather than focusing on acceptable worst-case bounds of security mechanisms, researchers have been overly concerned with minimizing their performance overhead.

In this paper, we have proposed a new security definition for database PHs. Any scheme that satisfies this definition allows for secure processing of encrypted data in the presence of an untrusted database service provider. Further, we have reasoned that this definition cannot be satisfied, suggesting that we should not expect too much from any previous or future work in this field.

On the bright side, we have given a general construction for a database PH that can be proved to be secure in a relaxed, but still rigorous and plausible sense under widely accepted cryptographic assumptions.

[2] One could model documents as sequences and not sets, but sets are strictly more general, and for our purposes the word order is irrelevant.

References

[1] D. Bleichenbacher. Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS#1. In *Advances in Cryptology*, 1998.

[2] C. Boyens and O. Günther. Trust is not Enough: Privacy and Security in ASP and Web Service Environments. In *ADBIS'02*, volume 2435 of *LNCS*.

[3] E. Damiani, S. De Capitani Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati. Balancing Confidentiality and Efficiency in Untrusted Relational DBMSs. In *CCS'03*. ACM Press.

[4] H. Hacıgümüş, B. Iyer, C. Li, and S. Mehrotra. Executing SQL over Encrypted Data in the Database-Service-Provider Model. In *SIGMOD'02*. ACM Press.

[5] M. Kantarcıoğlu and C. Clifton. Security Issues in Querying Encrypted Data. Purdue Computer Science Technical Report 04-013, 2004.

[6] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On Data Banks and Privacy Homomorphisms. In *Foundations of Secure Computation*. Academic Press, 1978.

[7] D. X. Song, D. Wagner, and A. Perrig. Practical Techniques for Searches on Encrypted Data. In *IEEE Symposium on Security and Privacy*, 2000.

Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)
8-7695-2570-9/06 $20.00 © 2006 IEEE